Introduction à l'optimisation de bases de données

BDD Avancées

Nicolas Travers
DUT Informatique

CNAM Paris
nicolas.travers (at) cnam.fr

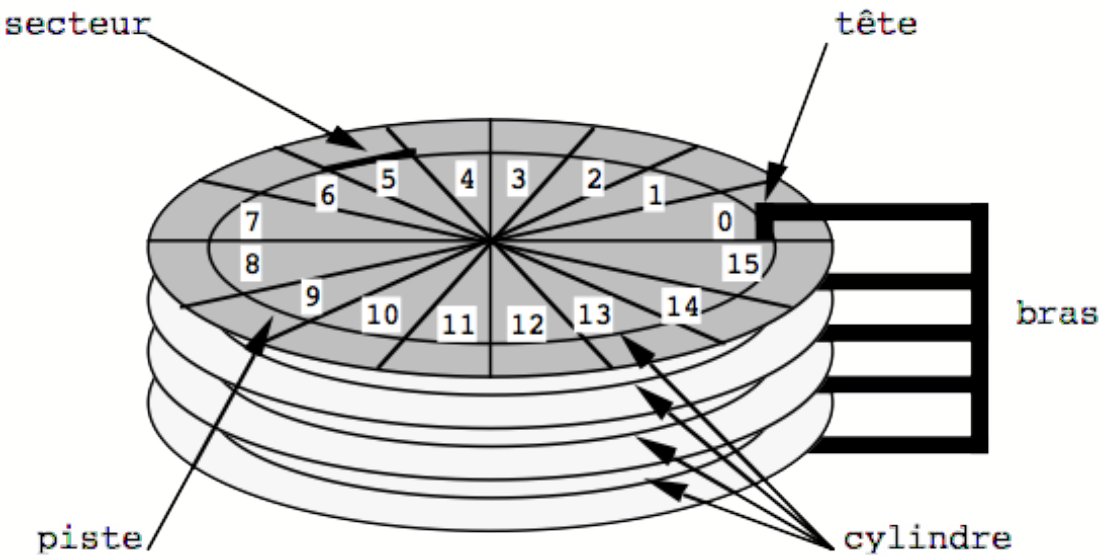
Plan

Qu'est-ce que l'optimisation dans un SGBD

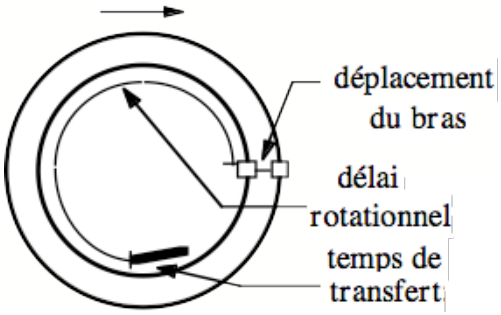
- Requête ⇒ Arbre d'opérateurs (algèbre relationnelle)
- Opérateur ⇒ Accès aux données
  - Coût en mémoire (place prise en mémoire)
  - Coût en accès aux données (I/O coûte cher)

⇒ **Minimiser les accès au disque**

Disque dur



Lecture d'un disque



- Délai rotationnel (rpm)

rpm	latency (ms)
15 000	2
10 000	3
7 200	4,16
5 400	5,55
4 800	6,25

- Temps total $\sim$ 9ms
  Dépend de :
  - Fragmentation des données
  - Changement de piste
- Taux de transfert : 125 Mo/s
  $\sim$ 300 Mo/s en SATA avec buffer

Entrées/Sorties

- Une entrée-sortie (I/O) $\Rightarrow$ Page disque
  - En anglais : *Block*
  - $\simeq$ Secteur sur le disque
  - $=$ 8 Ko (par défaut sous Oracle)
- Un fichier de données : plusieurs *pages* disques
- Plusieurs tuples par pages (dépend de la taille d'un tuple)

Exemple de stockage de *Cours*

Le **fichier** de *cours* peut être découpé de cette manière :

P1	2016-02-28	18h15	Travers	30	Introduction
	2016-03-15	18h15	Travers	32	Indexes
P2	2016-03-22	18h30	Travers	28	Optimisation
	2016-03-29	18h00	Hamdi	27	Dénormalisation
P3	2016-05-15	18h15	Travers	22	Tuning

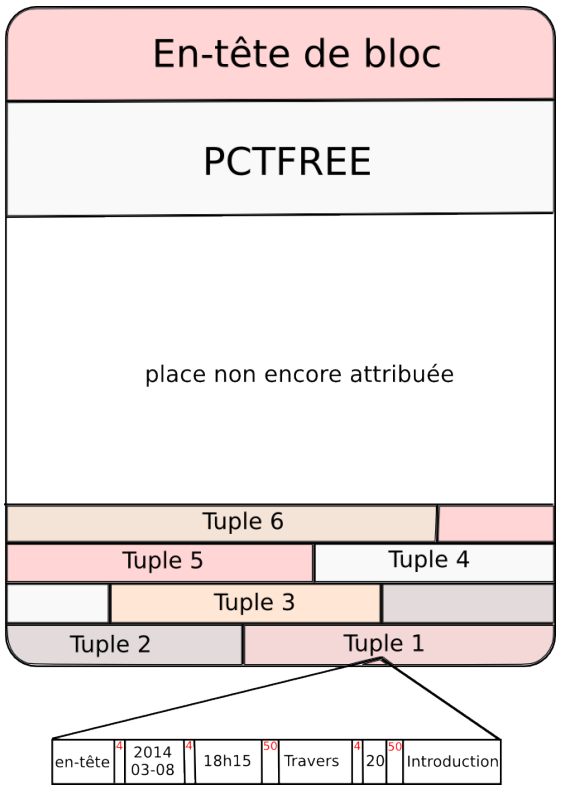
Page dique

Une page [1] contient :

- En-tête [2] :
  - Type de données (Relation, index, cluster...)
  - Schéma de données (attributs, null, blob...)
  - Pointeurs sur places vides, page suivante et précédente
- Place vide pour mises à jour
  - PCTFREE [3] / PCTUSED
- Les données
  - Tuples en entier
  - En-tête de tuple [4] (identification, valeurs NULL, verrous)
  - À chaque attribut s'associe sa taille [5]

1. Par défaut 8 ko : 8192 o
2. Entre 100 et 200 octets. Dans ce cours, nous utiliserons **192 o**
3. Par défaut 10%
4. Variable, mais en moyenne 3 octets
5. Taille de colonne entre 1 et 3 octets. En moyenne 1 octet

Pages et Taille d'une table



- Cours(Date, heure, nom, nb, intitule)
  5000 tuples
  Nb, Dates et heures : 4 o
  Texte : 50 o en moyenne.
- *Block* : 8 ko $\Rightarrow$ 8192 o
- $|DataBlock| = (8192 - 192) \times \frac{100 - PCTFREE}{100} =$ **7200 o**
- $|Tuple|$: $3 + 3 \times (4 + 1) + 2 \times (50 + 1) =$ **120 o**
- Nb tuples par page : $\left\lfloor \frac{7200}{120} \right\rfloor = 60 t/p$
- Nb pages : $\left\lceil \frac{5000}{60} \right\rceil = 84 pages$

Types de données

Types numériques

Type	Taille	Interval de valeurs
TINYINT	1 octet	0 à 255 ou -128 à 127
SMALLINT	2 octets	-32768 à 32767 ou 0 à 65535
MEDIUMINT	3 octets	-8388608 à 8388607 ou 0 à 16777215
INT, INTEGER	4 octets	-2147483648 à 2147483647 ou 0 à 4294967295
BIGINT	8 octets	$\sim 10^{19}$
FLOAT(p)	4 octets if $0 \leq p \leq 24$, 8 octets if $25 \leq p \leq 53$	nombre de décimales
FLOAT	8 octets	Par defaut, 53 décimales
DOUBLE(p), REAL	8 octets	
DECIMAL(M,D), NUM(M,D)	Variable	
BIT(M)	$\sim (M+7)/8$ octets	

Types Dates et Heures

Type	Taille
DATE	3 octets
TIME	3 octets
DATETIME	8 octets
TIMESTAMP	4 octets
YEAR	1 octet

Types textuel

Type	Taille
CHAR(M)	M octets ou ou M x 2 octets, $0 \leq M \leq 255$
BINARY(M)	M octets, $0 \leq M \leq 255$
VARCHAR(M)	M + 1o (0-255 o), M + 2o (> 255 o)
BLOB, TEXT	L + 2 octets, where $L \leq 215$
ENUM('v1','v2',...)	1o (255 valeurs), 2o (65 535 valeurs)
SET('v1','v2',...)	1, 2, 3, 4, or 8 octets (64 valeurs maximum)

Exemple simple d'accès

- $\pi_{Date}(\sigma_{Intitule='Introduction'}(Cours))$
  SELECT date FROM Cours WHERE Intitule='Introduction'
- Opérateur de sélection nécessaire.
  - Si pas d'index sur l'attribut *Intitule* :
  - Parcours chaque page (physique) de la relation *Cours* : `TABLE ACCESS FULL`
  - Si le tuple contient 'Introduction', le tuple est projeté
- **Coût de l'opération :** Nombre de pages de *Cours*

Notations

- **Tables :**
  - $|\mathcal{R}|$: Nb de pages de la relation $\mathcal{R}$
  - $|\mathcal{R}'|$: Nb de pages de la relation $\mathcal{R}$ après modification
  - $||\mathcal{R}||$: Nb de n-uplets de la relation $\mathcal{R}$
  - $||\mathcal{R}'||$: Nb de n-uplets de la relation $\mathcal{R}$ après sélection
- **Index :**
  - $|\mathcal{I}|$: Nb de pages à lire pour une tranversée d'index (ie. hauteur de l'arbre)
  - k : Ordre du l'arbre B
  - ϕ : *clustering factor* (cf. indexation)
  - *Sel* : Sélectivité d'un attribut
  - Δ : Nb de feuilles de $\mathcal{I}$ sélectionnées
- $|\mathcal{M}|$: Nb de pages tampons en MC pour la lecture

Plan

- Qu'est-ce qu'un index ?
- Index dense *versus* Index non dense
- Arbre-B+
- Index de hachage
- Index Bitmap

Principe d'indexation : Exemple de fichier

- Soit le fichier **Cours** contenant des données
- *Cours* (CODE, Intitule, Responsable, nb_auditeur)
- On souhaite accéder :
  - Au Responsable/Intitule d'un cours grâce à son CODE
  - Aux intitulés des cours d'un responsable
  - Aux nombre de cours ayant un certain nombre d'auditeurs

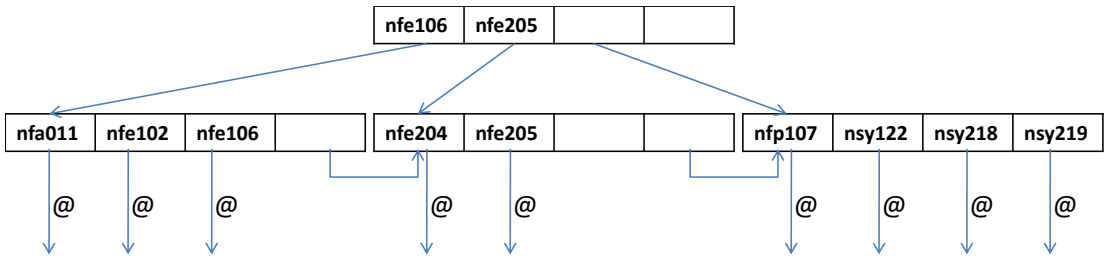
$\Rightarrow$ Besoins d'accès particuliers

Principe d'indexation : Exemple du fichier *Cours*

Page	CODE	Intitule	Responsable	NB_auditeurs
P1	NFE106	Ingénierie de Base de Données	Nicolas Travers	30
	NFP107	Initiation à la Base de Données	Michel Crucianu	100
	NFE204	Base de Données Avancées 1	Philippe Rigaux	30
P2	NFE205	Base de Données Avancées 2	Michel Crucianu	20
	NSY135	ORM et Framework	Philippe Rigaux	15
	NFA011	Approfondissement bases de données	Elisabeth Métais	55
P3	NFE102	Infrastructures Technologiques pour le Commerce Electronique	Philippe Rigaux	70
	NSY218	Vision par Ordinateur 2D	Valérie Gouet-Brunet	20
	NSY219	Vision par Ordinateur 3D	Valérie Gouet-Brunet	20

Fichier stocké sur 3 pages disques.

Exemple d'index



Principe d'indexation : Index

- Un index est stocké dans un autre fichier $\mathcal{I}$
- Besoin d'une clé d'accès $\mathcal{K}$: *CODE*
- Besoin d'un pointeur sur les données (ROWID) : @
- Caractéristiques :
  - $\mathcal{I}$ ne contient pas les données
  - A chaque $\mathcal{K}$ correspond un ou plusieurs ROWIDS @
  - $\mathcal{I}$ est ordonné sur les valeurs de $\mathcal{K}$
  - Entrée de $\mathcal{I}$: ($\mathcal{K}$, @)
  - Si $\mathcal{I}$ plus gros qu'une page, on indexe de la même manière chaque page de $\mathcal{I}$

Accès aux données

Deux étapes pour faire une recherche :

1. `INDEX RANGE SCAN` : Parcours de l'index
   - Donne l'adresse d'une page $\mathcal{P}$
   - On traverse l'arbre jusqu'aux feuilles de $\mathcal{I}$
   - Complexité : hauteur de l'arbre (Nb de pages)
   - Retour : liste de ROWIDS correspondant
2. `TABLE ACCESS BY ROWIDS` : Accès direct aux données
   - Ouverture de la page @
   - Parcours séquentiel de la page pour retrouver la(les) valeur(s) de $\mathcal{K}$

Index dense *versus* Index non dense

- **Index non dense**[6] : `Une page = un pointeur`
  - Valeurs ordonnées physiquement sur la clé $\mathcal{K}$
  - Une clé de l'index = premier tuple de chaque page
  - Très peu d'accès
  - Peu de place : $||\mathcal{K}|| = |\mathcal{R}|$
  - Un seul ordonnancement = un seul index non dense
- **Index dense** : `Une clé = un pointeur`
  - ROWID des feuilles très hétérogènes
  - Volumineux : $||\mathcal{K}|| = ||\mathcal{R}||$
  - Temps de parcours lent pour des valeurs multivaluées

⇒ Quel est le type de l'index précédent ?

⇒ Comment insérer un nouveau tuple dans l'index non dense ?

6. *Organization Index* (Oracle), *Clustered index* (SQLServer/DB2)

Arbre B+

- *Balanced Tree* : Arbre équilibré
- Implanté dans tous les SGBD relationnels (index par défaut)
- Les feuilles de l'arbre contiennent les pointeurs
- Principe des Arbres Binaires de Recherche

⇒ **Optimise les requêtes d'égalité**
(avec peu de valeurs, et les inéquations)

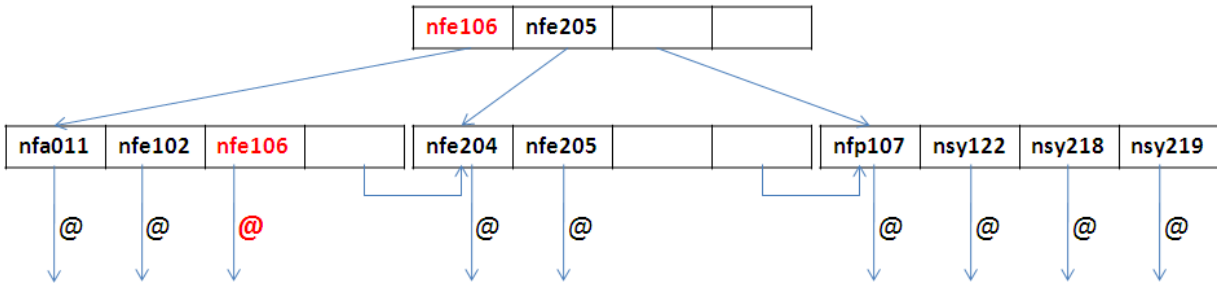
Arbre B+ : caractéristiques

- **Noeud** : clés + ROWID
- **Noeuds intermédiaires** : ROWID sur d'autres noeuds
- **Noeuds feuilles** : ROWID sur tuples du fichier indexé
- **Taille d'un noeud** : 2 * k valeurs (k est l'**ordre** de l'arbre)
- Quel est l'ordre de l'arbre de l'exemple ?

Arbre B+ : Recherche

- Recherche récursive de $\mathcal{K}$ à partir de la racine
- Soit C_1 à C_n les valeurs des clés de la page
  1. Si $\kappa \leq C_1$, recherche sur le noeud référencé P_1
  2. Si $\kappa > C_n$, recherche sur le noeud référencé P_{n+1}
  3. Si $C_i < \kappa \leq C_{i+1}$, recherche sur le noeud référencé P_{i+1}

Arbre B+ : Recherche de *nfe106*



Creation d'un BTree

- Btree dense :
  CREATE INDEX *Cours_nom* ON *Cours* (*nom*) ;
- Btree non-dense sous `Oracle` :
  CREATE TABLE Cours (...) *ORGANIZATION INDEX* ;

Hachage

- Partitionnement des données
- Très peu de place en mémoire
  - Fonction de hachage
  - Table d'adresses de partitions
- Collisions : plusieurs clés $\mathcal{K}$ par partitions
- Divise le temps de parcours par le nombre de partitions (en moyenne)

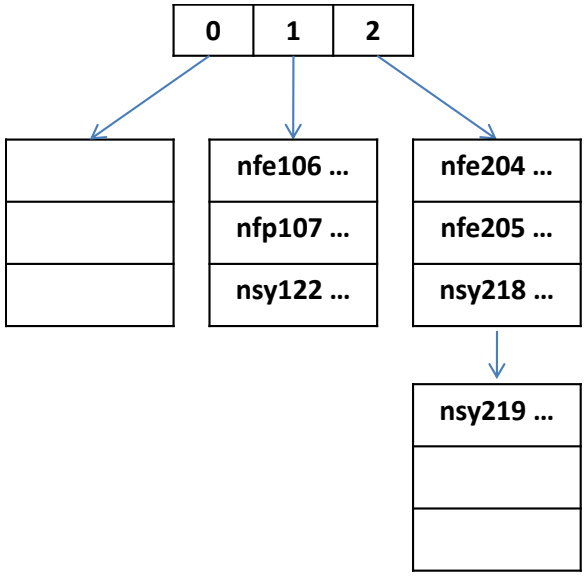
$\Rightarrow$ **Optimise les requêtes d'égalité**.
(Avec de nombreuses valeurs identiques)

Hachage : Caractéristiques

- **Fonction de hachage** : $h(\mathcal{K}) = \alpha$
  - $1 \leq \alpha \leq \beta$
  - β = Nb de partitions
  - h est une fonction uniforme[7] : $P(h(\kappa) = \alpha) = \frac{1}{\beta}$
  - Si $h(\mathcal{K}) = \alpha$, alors on ajoute la clé $\mathcal{K}$ (tout le n-uplet) à la page α
- **Table de hachage** T en MC[8]
  - $T[\alpha]$ = Adresse du bloc α
- Taille d'une partition : $\left\lceil \frac{|\mathcal{R}|}{\beta} \right\rceil$

7. Par défaut, *ORA_HASH* sous `Oracle`
`http://docs.oracle.com/cd/B28359_01/server.111/b28286/functions112.htm`
8. Mémoire Centrale

Hachage : Exemple



La clé de hachage prend la valeur numérique de l'UE.

Partitionnement

- Plusieurs types de partitionnement :
  - Hachage
  - Intervalles de valeurs
  - Liste de valeurs
  - Composition des précédentes
- Exemple de création :
  CREATE TABLE Cours (...)
  PARTITION BY HASH (RESPONSABLE) PARTITIONS 20 ;

Index : Récapitulatif

- **Requêtes très sélectives**
  - Arbre B+ dense (ie. clé primaire)
  - Arbre B+ non-dense (ie. beaucoup de données statiques ou incrémentales)
- **Requêtes sélectives**
  - Arbre B+ (ie. peu multivaluée)
  - Hachage (ie. beaucoup de valeurs)
- **Requêtes avec intervalle**
  - Arbre B+
- **Requêtes avec agrégats**
  - Bitmap

Opérateurs Relationnels

- Une requête SQL est transformée en Plan d'exécution
- Le plan est composé d'opérateurs
- Chaque opérateur à un coût d'évaluation
  - Coût en Entrées/Sorties
  - Dépend des statistiques (taille des tables, nb tuples, cardinalité, sélectivité...)

Sélection : Algorithmes

- Opération : $\sigma_{a=' x'}(\mathcal{R})$
- Sélection Séquentielle : `TABLE ACCESS FULL`
- Sélection avec Index Arbre B+ : `INDEX RANGE SCAN`
- Sélection par Hachage : `PARTITION HASH SINGLE`

TABLE ACCESS FULL : Sélection Séquentiel

- Appliquer si aucun index sur $\mathcal{R}(a)$
- Pour chaque page b de $\mathcal{R}$
  - Pour chaque tuple t de b
    - Si $t.a =' x'$
      Ajouter t dans $\mathcal{S}$

$\Rightarrow$ Complexité ? $|\mathcal{R}|$

TABLE ACCESS FULL : $\sigma_{Prof='MichelCrucianu'}(Cours))$

Page	CODE	Intitule	Responsable
P1	NFE106	Ingénierie de Base de Données	Nicolas Travers
	NFP107	Initiation à la Base de Données	Michel Crucianu
	NFE204	Base de Données Avancées 1	Michel Scholl
P2	NFE205	Base de Données Avancées 2	Michel Crucianu
	NSY122	Analyse des images et des sons numériques	Valérie Gouet-Brunet
	NFA011	Approfondissement bases de données	Elisabeth Métais
P3	NFE102	Infrastructures Technologiques pour...	Francois-Yves Villemin
	NSY218	Vision par Ordinateur 2D	Valérie Gouet-Brunet
	NSY219	Vision par Ordinateur 3D	Valérie Gouet-Brunet

Tampon de sortie

CODE	Intitule	Responsable
NFP107	Initiation à la Base de Données	Michel Crucianu

Opérations Plans Intro Indexes Opérations Sel

TABLE ACCESS FULL : $\sigma_{Prof='MichelCrucianu'}(Cours))$

Page	CODE	Intitule	Responsable
P1	NFE106	Ingénierie de Base de Données	Nicolas Travers
	NFP107	Initiation à la Base de Données	Michel Crucianu
	NFE204	Base de Données Avancées 1	Michel Scholl
P2	NFE205	Base de Données Avancées 2	Michel Crucianu
	NSY122	Analyse des images et des sons numériques	Valérie Gouet-Brunet
	NFA011	Approfondissement bases de données	Elisabeth Métais
P3	NFE102	Infrastructures Technologiques pour...	Francois-Yves Villemin
	NSY218	Vision par Ordinateur 2D	Valérie Gouet-Brunet
	NSY219	Vision par Ordinateur 3D	Valérie Gouet-Brunet

Tampon de sortie

CODE	Intitule	Responsable
NFP107	Initiation à la Base de Données	Michel Crucianu
NFE205	Base de Données Avancées 2	Michel Crucianu

TABLE ACCESS FULL : $\sigma_{Prof='MichelCrucianu'}(Cours)$

Page	CODE	Intitule	Responsable
P1	NFE106	Ingénierie de Base de Données	Nicolas Travers
	NFP107	Initiation à la Base de Données	Michel Crucianu
	NFE204	Base de Données Avancées 1	Michel Scholl
P2	NFE205	Base de Données Avancées 2	Michel Crucianu
	NSY122	Analyse des images et des sons numériques	Valérie Gouet-Brunet
	NFA011	Approfondissement bases de données	Elisabeth Métais
P3	NFE102	Infrastructures Technologiques pour...	Francois-Yves Villemin
	NSY218	Vision par Ordinateur 2D	Valérie Gouet-Brunet
	NSY219	Vision par Ordinateur 3D	Valérie Gouet-Brunet

Tampon de sortie

CODE	Intitule	Responsable
NFP107	Initiation à la Base de Données	Michel Crucianu
NFE205	Base de Données Avancées 2	Michel Crucianu

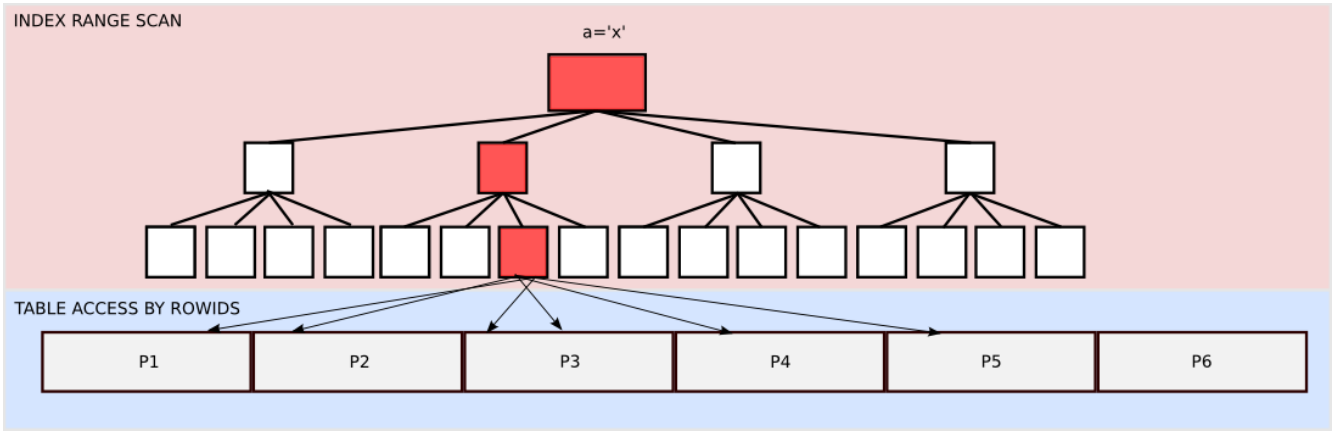
INDEX RANGE SCAN + TABLE ACCESS BY INDEX ROWID

- Sélection par index appliqué si $\mathcal{I}$ sur $\mathcal{R}(a)$
- `INDEX RANGE SCAN`
  - Ouvrir la page $\mathcal{P}$ racine de $\mathcal{I}$
    Tant que $\mathcal{P}$ n'est pas une feuille
    Parcourir l'arbre de $\mathcal{P}$ avec a='x'
    $\mathcal{P}$ = nouvelle page cible
  - Sortie : $\mathcal{L}$ = liste de ROWIDS
    - Selectivité de l'attribut a : Sel
    - Pourcentage de sélection : $0 < Sel < 1$

    $\Rightarrow$ $||\mathcal{L}|| = ||ROWID|| \times Sel$
- `TABLE ACCESS BY INDEX ROWID`
  - Pour chaque l de $\mathcal{L}$
    Ouvrir la page p pointée par l
    Pour chaque tuple t de p
    Si $t.a =$ 'x'
    Ajouter t dans le tampon de sortie

$\Rightarrow$ Complexité : $|\mathcal{I}| + ||ROWID|| \times Sel$

INDEX RANGE SCAN + TABLE ACCESS BY INDEX ROWID : Exemple index dense



INDEX RANGE SCAN : Sélectivité

- **Pourcentage** de ROWID pour une valeur cherchée
- Dépend de la répartition des données de l'attribut
- Par défaut :
  - $Sel(\mathcal{R}\ (a)) = \frac{1}{Card(\mathcal{R}(a))}$ [9]
  - Exemple :
    - $\mathcal{R}\ (a)$ est réparti sur 20 valeurs différentes (cardinalité = 20)
    - Statistiquement : $Sel(\mathcal{R}\ (a)) = \frac{1}{20} = 5\%$
  - Clé primaire : $Sel = \frac{1}{||\mathcal{R}||}$ (INDEX *UNIQUE* SCAN)
- Statistiques
  - Histogramme de fréquences par valeur ($\mathcal{R}$(a))

NFA011	NFE102	NFE106	NFE204	NFE205	NFP107	NSY218
5%	2%	6%	4%	3%	2.5%	1.2%

⚠ Coût de calcul / mise à jour

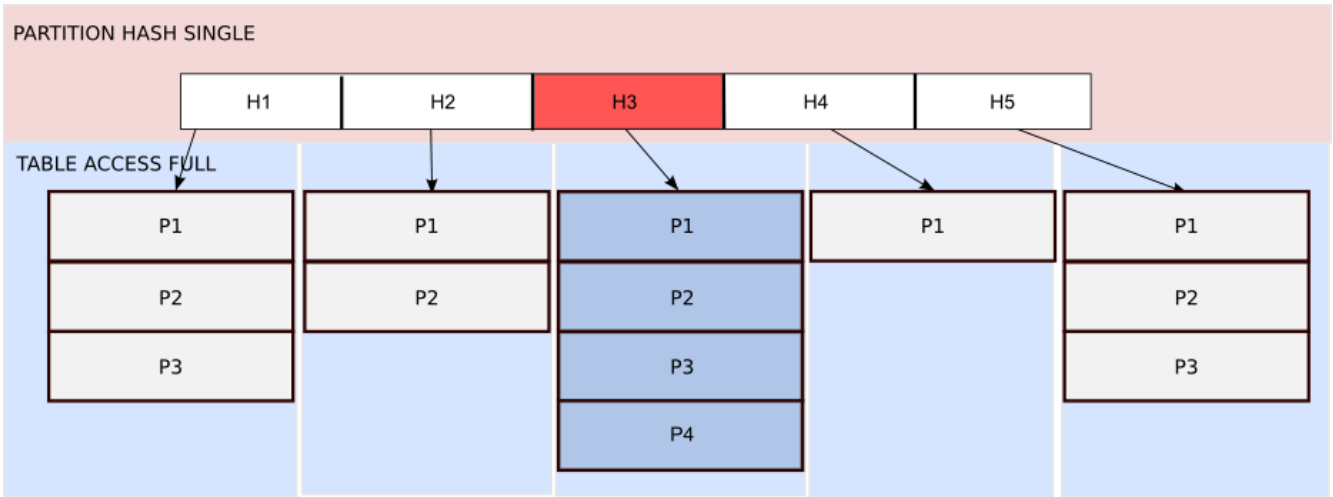
9. $Card(\mathcal{R}(a))$: nb de valeurs distinctes

PARTITION HASH SINGLE : Sélection par Hachage

- Appliquer si index Hash $\mathcal{H}$ sur $\mathcal{R}(a)$
- $\mathcal{P}$ partitions
- Données placées sur les partitions en fonction de $\mathcal{H}$
- Hacher la clé $'x'$: $\mathcal{H}('x') = v$ $(1 \leq v \leq \mathcal{P})$
- Recherche :
  Pour chaque page p de la partition $\mathcal{R}(v)$
  - Pour chaque tuple t de p
    - Si $t.a =' x'$
      Ajouter t dans $\mathcal{S}$

$\Rightarrow$ Complexité ? Taille d'une partition $\approx \left\lceil \frac{|\mathcal{R}|}{\mathcal{P}} \right\rceil$

PARTITION HASH SINGLE : Exemple



PARTITION HASH SINGLE : Partitionnement

- Répartition des clés en fonction de la fonction de hachage [10]
- N-uplets uniforméments répartis sur les $\mathcal{P}$ partitions

⚠ Problème de répartition

- Fonction inadaptée
- Répartition inégale des redondances (reste meilleur qu'un Arbre B+)

⇒ Combinaison de hachage

⇒ Partitionnement par intervalles

10. Sous Oracle : `ORA_HASH`. Peut être surchargée

Introduction

- **SQL est déclaratif**
  - On dit ce que l'on veut obtenir
  - On ne dit pas comment l'obtenir
- **Optimiseur** doit :
  - Comprendre la requête. Traduction en **P***lan d'***E***xécution* **L***ogique*[11] (PEL)
  - Choisir le meilleur **P***lan d'***E***xécution* **P***hysique*[12] (PEP)
  - Exécuter le PEP

11. Opérateurs de l'algèbre relationnelle
12. Opérateurs implémentés dans le SGBD

Introduction : Exemple

- Soit le schéma :
  - *Cours* (CODE, Intitule, Responsable)
  - *Auditeurs* (CODE_UE, CODE_AUDITEUR, Annee, Note)
- Soit la requête :
  - Liste des *années* où *Nicolas Travers* a eu des auditeurs ?

⇒
```
SELECT Annee
FROM Cours, Auditeurs
WHERE Responsable = 'Nicolas Travers' AND
        CODE = CODE_UE;
```

Introduction : Exemple PEL

1. $\pi_{Annee}(\sigma_{Responsable='NicolasTravers'}(Cours \bowtie_{CODE=CODE\_UE} Auditeurs))$
2. $\pi_{Annee}(\sigma_{Responsable='NicolasTravers'}(Cours) \bowtie_{CODE=CODE\_UE} Auditeurs)$
3. $\pi_{Annee}(\sigma_{Responsable='NicolasTravers'}(Auditeurs \bowtie_{CODE=CODE\_UE} Cours))$
4. $\pi_{Annee}(Auditeurs \bowtie_{CODE=CODE\_UE} \sigma_{Responsable='NicolasTravers'}(Cours))$

Quel est le meilleur plan d'exécution ?

$\Rightarrow$ Le second PEL

EXPLAIN : Outils d'affichage de PEP

- Outils de représentation de PEP sous Oracle et MySQL
- Représentation arborescente (tabulations)
- Estimation du coût d'évaluation
- Insertion de statistiques d'évaluation

EXPLAIN : Exemple

Id	Opération	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY ROWIDS	Cours
2 *	INDEX RANGE SCAN	Cours_NB_AUDITEUR_IDX

(2) : access(NB_AUDITEUR BETWEEN 20 AND 30)

Table EXPLAIN PLAN (Oracle)

- Toutes les données sont stockées ⇒ Plans `EXPLAIN` aussi

Oracle Commande de transformation SQL ⇒ `EXPLAIN` :

- EXPLAIN PLAN [SET STATEMENT_ID = 'ID'][13]
  [INTO < table_name >][14]
  FOR < Requête SQL >
- Commande d'affichage automatique du plan :
  `SET AUTOTRACE ON`

MySQL Commande d'affichage de EXPLAIN : `EXPLAIN` < Requête SQL >

13. Identifiant pour retrouver le plan

Sélection / Accès aux données

Opération	Option	Description
`TABLE ACCESS`	`FULL`	Sélection séquentielle
	`BY INDEX ROWID`	Accès aux ROWIDS
	`CLUSTER`	Accès via la clé d'index de cluster

Accès à un Index

Opération	Option	Description
INDEX	UNIQUE SCAN	clé primaire/unique
	RANGE SCAN	données multivaluées
	FULL SCAN	toutes les clés de l'index
	FAST FULL SCAN	toutes les clés de l'index sans prendre les ROWIDS
	FULL SCAN (MIN/MAX)	min/max des valeurs de l'index

Accès à un Index Hash

Opération	Option	Description
PARTITION HASH	SINGLE	Accès à la partition sélectionné par la clé d'accès (sur le FULL SCAN)
	ALL	Accès à toutes les partitions (pas de filtrage sur la clé hachée)

Accès à un Index Bitmap

Opération	Option	Description
BITMAP	INDEX SINGLE VALUE	Accès à l'index bitmap, retourne les numéros des tuples
	CONVERSION COUNT	Compte le nombre de bit à 1
	CONVERSION TO ROWIDS	Converti les numéros de tuples en ROWIDs
	AND	ET logique entre deux listes bitmap
	OR	OU logique entre deux listes bitmap
	MINUS	AND NOT logique
	INDEX RANGE SCAN MERGE	Accès à un ensemble de listes Fusionne les listes retournées par un RANGE SCAN

Tri vs Hachage

Opération	Option	Description
SORT	ORDER BY	*Sort* pour les résultats
	AGGREGATE	Tri avec fonction d'aggrégat [15]
	GROUP BY	Tri sans fonction (juste Group By)
	UNIQUE	DISTINCT
	JOIN	Étape de tri de l'algo de jointure

Opération	Option	Description
HASH	AGGREGATE	Hachage avec fonction d'aggrégat
	GROUP BY	Hachage sans fonction
	UNIQUE	DISTINCT

15. avec ou sans Group By

Jointures

Opération	Description
`NESTED LOOPS`	*NLJ*/*NLJI*, Boucles imbriquées
`MERGE JOIN`	*SMJ*, Etape de fusion des tables
`HASH JOIN`	*HashJoin*, construction des partitions, puis suite de *NLJ*
`AND-EQUAL`	Intersection de deux listes de ROWIDs pour une jointure entre deux indexes

Options : `OUTER, ANTI, SEMI, CARTESIAN`.

Bonus (1/3)

Opération	Description
`CONNECT BY`	Auto-jointure hiérarchique utilisant le résultat de l'étape précédente
`CONCATENATION`	Union multiples de *Result Sets* (cf `UNION`)
`COUNT`	Compte le nombre tuples d'un résultat
`FILTER`	Opération de sélection
`FIRST ROW`	Premier tuple sélectionné par la requête
`FOR UPDATE`	Blocage des tuples retournés pour des MAJ ultérieures

Bonus (2/3)

Opération	Description
`VIEW`	Accès à une vue, ou création d'une table temporaire
`UNION`	Union de deux *Result Set*, sans doublons
`UNION-ALL`	Union de deux *Result Set*, avec doublons
`INTERSECTION`	Intersection de deux *Result Set*, contenant les mêmes valeurs
`MINUS`	Les tuples du premier *Result Set* sont retournés, sauf ceux présents dans le second

Bonus (3/3)

Opération	Description
`SEQUENCE`	Oracle Sequence Generator (auto_increment d'ID)
`REMOTE`	Accès à une base de données distante
`INLIST OPERATOR`	Un seul opérateur à la fois. Pas de *pipeline*

Lecture de EXPLAIN

- Structure arborescente (indentation)
- L'opérateur le moins indenté est le résultat
- Le plus indenté est l'accès aux données
- Si deux indentation de même niveau, le premier est exécuté d'abord
- Exercice sur les plans suivants :
  - Requête SQL correspondante
  - Coût d'évaluation

Exemple 1 : Sélection séquentielle

Id	Opération	Name
0	SELECT STATEMENT	
1 *	TABLE ACCESS FULL	Auditeurs

(1) : access(Annee = 2016)

- SELECT ANNEE FROM Auditeurs WHERE ANNEE=2016 ;

Exemple 2 : Accès via Index unique

Id	Opération	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY ROWIDS	Cours
2 *	INDEX UNIQUE SCAN	Cours_Code_IDX

(2) : access(CODE = 'NFE106')

- SELECT RESPONSABLE FROM Cours WHERE CODE='NFE106' ;

Exemple 3 : Accès via Index dense

Id	Opération	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY ROWIDS	Cours
2 *	INDEX RANGE SCAN	Cours_NB_AUDITEUR_IDX

(2) : access(NB_AUDITEUR BETWEEN 20 AND 30)

- SELECT RESPONSABLE FROM Cours
  WHERE NB_AUDITEUR BETWEEN 20 AND 30 ;

Exemple 4 : Accès via Table de Hachage

Id	Opération	Name
0	SELECT STATEMENT	
1	PARTITION HASH SINGLE	
2 *	TABLE ACCESS FULL	Cours

(2) : access(Responsable = 'Nicolas Travers')

- SELECT CODE FROM Cours WHERE Responsable = 'Nicolas Travers' ;

EXPLAIN : Statistiques

- EXPLAIN fourni des informations statistiques
  - Nombre d'appels physiques
  - Nombre d'accès mémoires
  - Traitements temporaires

⇒ Permet de comprendre le coût d'évaluation du plan

Exemple de statistiques

Id	Opération	Name
0	SELECT STATEMENT	
1 *	NESTED LOOPS	
2	TABLE ACCESS FULL	Cours
3	TABLE ACCESS FULL	Auditeurs

(1) : filter(CODE = CODE_UE)

Statistics
0 recursive calls
100 db block gets
2000 consistent gets
300 physical reads

Différentes Statistiques

- Accès mémoire :
  - **consistent gets** : Nb d'accès à une donnée consistente en RAM (non modifiée)
  - **db block gets** : Nb d'accès à une donnée en RAM
- Accès physiques :
  - **physical reads** : Nb total de pages lues sur le disque
- **recursive calls** : Nb d'appels à un sous-plan (requête imbriquée)
- **redo size** : Taille du fichier de log produit (update)
- **sorts (disk)** : Nb d'opérations de tri avec au moins une écriture
- **sorts (memory)** : Nb d'opérations de tri en mémoire

$\Rightarrow$ Taux d'efficacité du cache : $1 - \frac{physical\ \ reads}{consistent\ \ gets\ +\ db\ \ block\ \ gets}$